



iPerl - next generation REPL for Perl

Árpád Szász
Freelance Perl Developer
Twitter: @arpadszasz
Blog: <http://arpi.plenum.ro>

Introduction

Introduction

- iPerl is a GUI based REPL

Introduction

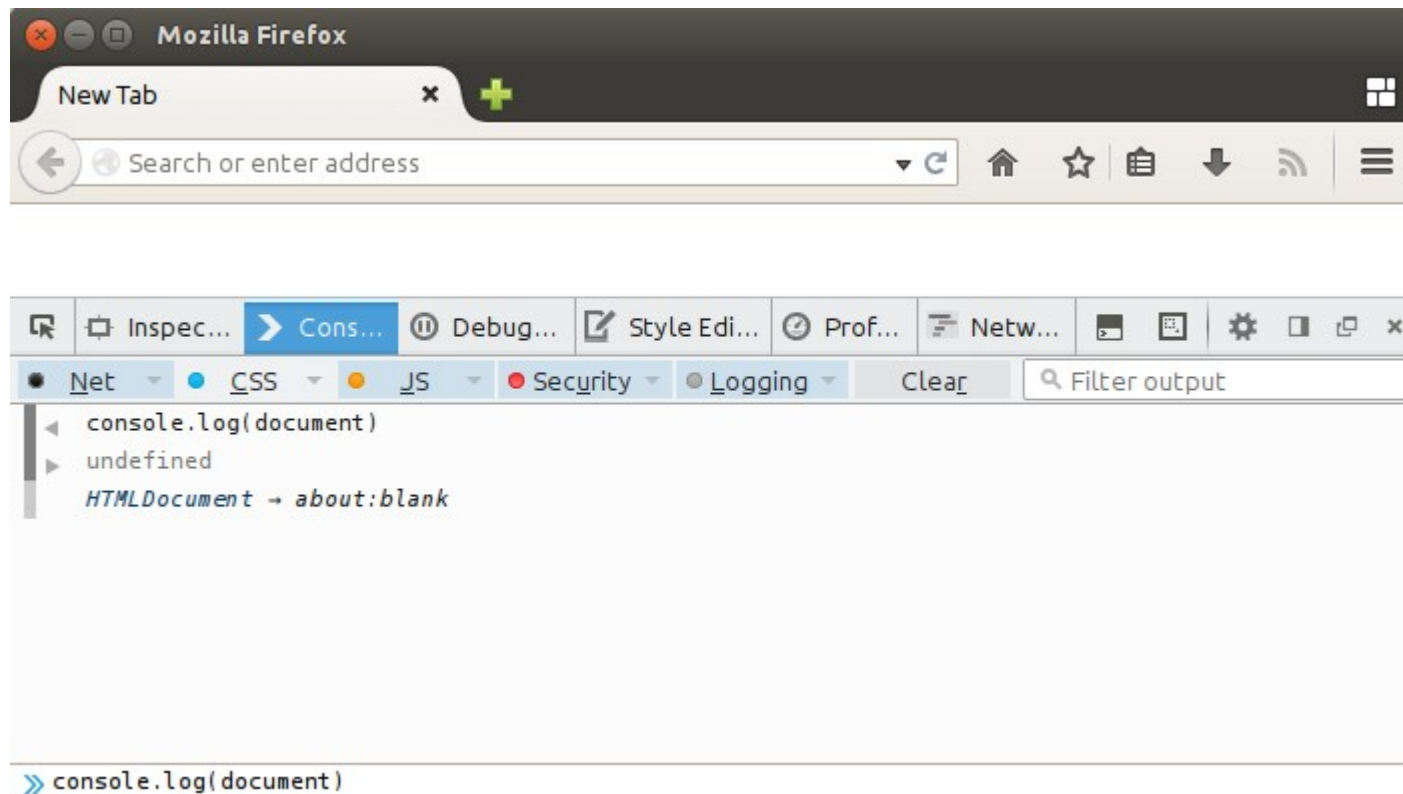
- iPerl is a GUI based REPL
- iPerl has live-coding features

REPL

- Read-Eval-Print-Loop

REPL

- Read-Eval-Print-Loop



REPL

- CPAN already has a lot of REPLs

REPL

- CPAN already has a lot of REPLs

meta::cpan

REPL

Search

104 results (0.12 seconds)

SOMETHING MISSING?
[Find out why](#)

Carp::REPL - read-eval-print-loop on die and/or warn 2 ++
TSIBLEY/Carp-REPL-0.16 - May 29, 2013 - [Search in distribution](#)

App::REPL - A container for functions for the iperl program ++
AYRNIEU/App-REPL-0.012 - Apr 11, 2007 - [Search in distribution](#)

Repl::Loop - A command interpreter for applications. ++
It is a read-eval-print loop that supports a lisp-like syntax. It is meant as the top-level for command line oriented ap
BRANSCHA/Repl-Loop-1.00 - May 22, 2010 - [Search in distribution](#)
[Repl::Spec::Types](#) - A module with common type guards.
[Repl::Spec::Type::BooleanType](#) - A parameter guard for boolean values.
[Repl::Spec::Type::InstanceType](#) - A parameter guard that ensures instances of a specified class.
[37 more results from Repl-Loop »](#)

File::Repl - Perl module that provides file replication utilities 1 ++
The File:Repl provides simple file replication and management utilities. Its main functions are File Replication Allow
DROBERTS/File-Repl-1.31 - Jan 26, 2014 - [Search in distribution](#)

Devel::REPL - a modern perl interactive shell 36 ++
This is an interactive shell for Perl, commonly known as a REPL - Read, Evaluate, Print, Loop. The shell provides :

REPL

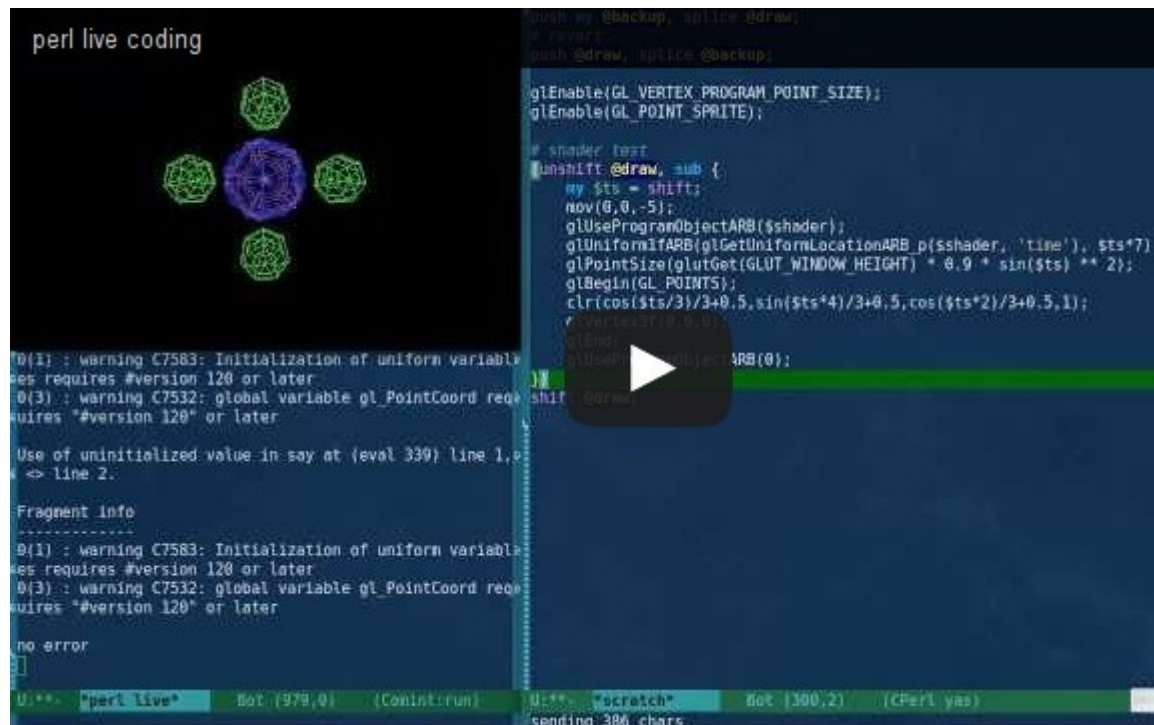
- Features of iPerl as a REPL
 - GUI based

REPL

- Features of iPerl as a REPL
 - GUI based
 - Simple to install and use

Live coding

- Idea and demo by vividsnow on blogs.perl.org



Live coding

- Idea and demo by vividsnow on blogs.perl.org
 - Emacs perl-live mode

Live coding

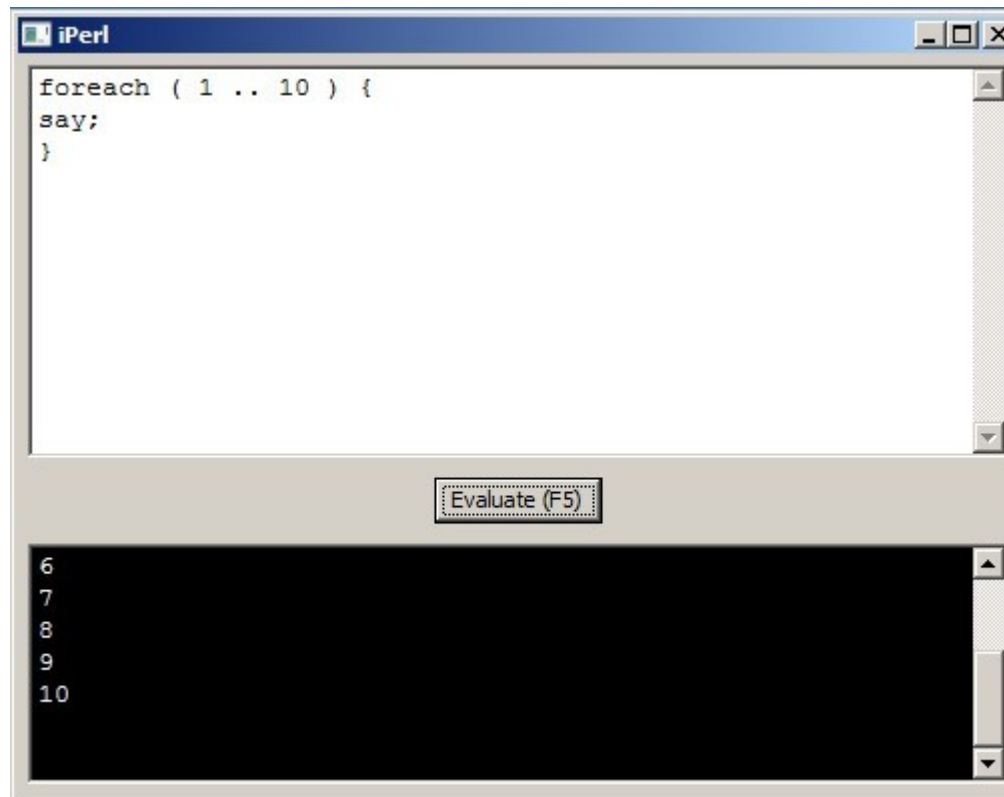
- Idea and demo by vividsnow on blogs.perl.org
 - Emacs perl-live mode
 - AnyEvent loop

Live coding

- Idea and demo by vividsnow on blogs.perl.org
 - Emacs perl-live mode
 - AnyEvent loop
 - PadWalker + Package::Stash

Current status of iPerl

- It works



The screenshot shows a web browser window titled "iPerl". The main text area contains the following Perl code:

```
foreach ( 1 .. 10 ) {  
  say;  
}
```

Below the code area is a button labeled "Evaluate (F5)". The output area, which has a black background, displays the numbers 6, 7, 8, 9, and 10, each on a new line.

Current status of iPerl

- It works
 - wxPerl based GUI

Current status of iPerl

- It works
 - wxPerl based GUI
 - 1 thread for GUI, 1 thread for eval-ing code

Current status of iPerl

- It has bugs!

Current status of iPerl

- It has bugs!
 - 'exit' stops the whole program

Current status of iPerl

- It has bugs!
 - 'exit' stops the whole program
 - 'exec', 'sleep', 'while' will hang the whole program

Current status of iPerl

- It has bugs!
 - 'exit' stops the whole program
 - 'exec', 'sleep', 'while' will hang the whole program
 - More unknown bugs!

Current status of iPerl

- It has bugs!
 - 'exit' stops the whole program
 - 'exec', 'sleep', 'while' hang the whole program
 - More unknown bugs!
 - ALPHA level quality!

Future plans for iPerl

- General plans
 - Syntax-highlighting

Future plans for iPerl

- General plans
 - Syntax-highlighting
 - Code auto-complete

Future plans for iPerl

- General plans
 - Syntax-highlighting
 - Code auto-complete
 - Colorized output

Future plans for iPerl

- General plans
 - Syntax-highlighting
 - Code auto-complete
 - Colorized output
 - Restrict “un-safe” commands (Safe.pm)

Future plans for iPerl

- General plans
 - Syntax-highlighting
 - Code auto-complete
 - Colorized output
 - Restrict “un-safe” commands (Safe.pm)
 - Load/save edited code

Future plans for iPerl

- Advanced plans
 - Replace subroutines on-the-fly

Future plans for iPerl

- Advanced plans
 - Replace subroutines on-the-fly
 - symbol table manipulation

Future plans for iPerl

- Advanced plans
 - Replace subroutines on-the-fly
 - symbol table manipulation
 - optree manipulation using B::modules (EVIL!)

Future plans for iPerl

- Advanced plans
 - Replace variables on-the-fly

Future plans for iPerl

- Advanced plans
 - Replace variables on-the-fly
 - “visible” package variables (mostly safe)

Future plans for iPerl

- Advanced plans
 - Replace variables on-the-fly
 - “visible” package variables (mostly safe)
 - lexical “private” variables (EVIL!)

**Code contributions/ideas
welcome!**

<http://bit.ly/iperl>

Thank You!